An End-to-end Approach for Learning and Generating Complex Robot Motions from Demonstration

Ali H. Kordia[1] Francisco S. Melo[2]

***Abstract*— This paper proposes an end-to-end framework that can learn and decompose complex movements provided by a human demonstrator and generate new complex motions. Our approach analyzes the demonstration by the human expert and uses geometric criteria to decompose the observed movement into segments that are stored as dynamic movement primitives (DMPs) in a library. Then, given a new environment configuration, our system autonomously composes different motion primitives to construct an optimized trajectory that meets the constraints imposed by the new environment. Our system is therefore able to construct new DMPs to execute complex motions in environments that differ from the one where the original motions were taught. Our approach is also compatible with existing run-time obstacle avoidance approaches. We illustrate the application of our approach both in simulation and with a Baxter robot.**

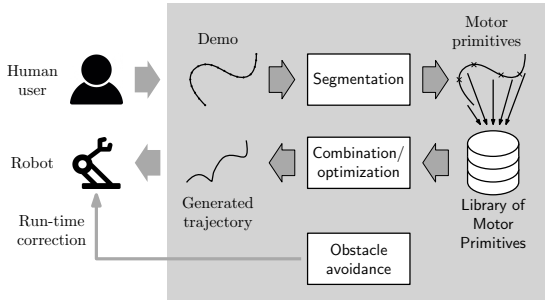


Fig. 1: Overview of the proposed system architecture for end-to-end learning, decomposing, optimizing and combining robot motions.

I. Introduction

As robots become a more common reality, their applicability broadens. As a consequence, it becomes increasingly more difficult to pre-program a robot to deal with the multitude of contexts and tasks that it will face when deployed. The ability of a robot to learn new tasks from demonstration is, therefore, a fundamental skill that robots should necessarily possess [1].

There is a vast literature on learning from demonstration [2], [3]. It is possible, for example, to consider learning from demonstration at the *task level*, where the learner seeks to infer the goal of the demonstration and adopt it as its own [4], [5]. Conversely, it is possible to address learning from demonstration at the motor resonance level [6], [7], where the learner simply seeks to replicate the motor behavior observed from the demonstration. Our work lies somewhere in the middle, and addresses learning from demonstration as learning how to perform and combine "atomic actions", referred as *sensory-motor primitives*—or, when disregarding perception issues, simply as *motor primitives* [8], [9].

In particular, we use demonstrations from the user to build a library of abstract atomic actions (the aforementioned motor primitives) that are then combined to perform complex motions. We adopt the formalism of *dynamic motor primitives* (DMPs) [10] to represent such abstract atomic actions. A DMP consists of a parameterized dynamical system whose parameters are adjusted in such a way that the trajectories of the system closely replicate those observed in the demonstration. [1] DMPs are robust to perturbations and can be easily learned from demonstrations. Additionally, modulation (i.e., performing the same motion to different end-points) can easily be achieved by changing some of the driving terms of the underlying dynamical system.

This paper proposes an end-to-end approach to learning and decomposing user demonstrations into DMPs, and then optimizing and combining the resulting DMPs to perform complex motions. Our approach is summarized in Fig. 1. The use DMPs greatly facilitates the different components of our architecture, since they are amenable to optimization [11], can easily be combined to generate complex motions [12], and can be exploited to ensure the avoidance of obstacles in runtime [13], [14]. Our framework comprises a library of DMPs and a set of different modules for (i) segmenting a demonstration provided by a human user, abstracting the computed sub-trajectories into new DMPs and adding them to the library; (ii) a greedy algorithm that, given a goal point, combines the DMPs in the library into a single complex motion that is then optimized according to provided criteria (length of trajectory, obstacle avoidance, etc.); (iii) a module that, during execution, monitors the existence of obstacles and reshapes the trajectory to avoid them in run-time.

II. Related work

Our work builds on—or is closely related to—different existing approaches that address each of the components of our framework individually. Our main contribution is, therefore, the combination of these different modules into a unified end-to-end framework for robot motion segmentation, representation, combination and optimization.

[1]Ali Hani Kordia is with INESC-ID and with Instituto Superior Técnico, University of Lisbon, Portugal; ali.kordia@tecnico.ulisboa.pt.

[2]Francisco S. Melo is with INESC-ID and with Instituto Superior Técnico, University of Lisbon, Portugal; fmelo@inesc-id.pt.

[1]Note that DMPs are a way to *represent* a trajectory, and not a direct control approach.

Given a demonstration by a human user, several approaches have considered the problem of identifying *sub-trajectories* that, in themselves, are sufficiently distinct as to be considered different motor primitives. For example, given a library of DMPs, it is possible to segment complex movements into a sequence of simpler movements by recognizing individual segments of the larger trajectory as instances of the movements in the library [15], [16]. The authors then use an expectation-maximization algorithm to determine both the endpoints of the segments and to recognize which individual DMPs are present in the observed trajectory.

In another work [17], the authors propose the use a Bayesian non-parametric model (an auto-regressive hidden Markov model) to describe the observed demonstration, and use the estimated latent discrete states to segment the trajectory into different sub-trajectories. Each such sub-trajectory is then represented as a DMP and used to build a library that can latter be used for planning.

Our approach also uses the trajectories provided by the demonstrator to build a library of DMPs that can then be used for planning. Our segmentation approach builds on that proposed by Meier et al [16], in that we use velocity and acceleration information to determine points of segmentation.

As for motion combination, it can be perceived from at least two distinct perspectives. On one hand, motions can be generated by *mixing* motor primitives. For example, Mulling et al [18] propose an approach that learns how to build mixtures of motor primitives to more effectively hit a moving table tennis ball. On the other hand, complex motions can be built by *sequencing* motor primitives [19], [12] build. For example, Kulvicius et al [12] propose a simple approach in which two DMPs can be smoothly sequentially combined by adding a common term that drives both DMPs in the vicinity of the transition points.

Finally, motion optimization is a vast topic that includes contributions ranging from reinforcement learning [20], [21] to black-box optimization [22], [23], [24]. In our approach, we adopt the CMA-ES (Covariance Matrix Adaptation Evolutionary Strategy) algorithm for trajectory optimization [15], but refer to the discussion in [25] for a comparative view of different approaches. We use CMA-ES as it is an efficient local search method that makes few assumptions on the domain and objective function to be optimized [26].

III. A Framework for Motion Decomposition, Learning and Generation

Our framework is summarized in Fig. 1 and comprises several modules for motion decomposition, learning and combination. Underlying all the modules is a common representation of "atomic" robot motions—the dynamic motion primitive, or DMP.

DMPs consist of a stable nonlinear dynamical system (a damped spring system, see Fig. 2 driven by a force term that is responsible for shaping the trajectory [10]. Each degree of freedom of the robot is controlled by an individual DMP, and synchronization is achieved by one additional dynamical component known as the *canonical system*. The use a canonical system instead of a synchronized clock ensures that the resulting system is time-homogeneous, which in turn brings advantages in terms of control.

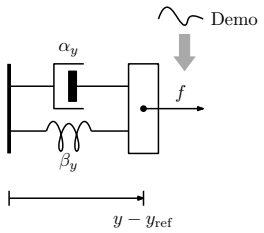


Fig. 2: Depiction of the dynamical system used as the basic representation of a DMP. A DMP is a damped spring system, where the state, y, represents the quantity of interest (robot position, joint position, etc.). The forcing function f is used to control the shape of the trajectory to match that provided by an expert demonstration.

The forcing function is usually selected as belonging to some parametric family, with the parameters tuned to yield the motion that the DMP is expected to represent (see Fig. 2).

Formally,[2] a DMP is thus a dynamical system

$$\ddot{y}(t) = \alpha_y(\beta_y(y_{\text{ref}} - y(t)) - \dot{y}(t)) + f(x(t)), \qquad (1)$$

where $y(t)$ is the *state* of the DMP, α_y and β_y are scalar gains, y_{ref} is the reference position (i.e., the target position for the system), f is the force term and $x(t)$ is the state of the canonical system. Throughout this work, we use a *sigmoidal canonical system* [12]: for a trajectory of length T, the canonical system is defined as

$$\dot{x}(t) = -\frac{\alpha_x \exp\{\alpha_x(T-t)/\Delta t\}}{1 + \exp\{(\alpha_x(T-t)/\Delta t)^2\}}$$

where α_y^s controls the steepness of the sigmoidal function and Δt is the sampling rate. The system in (1) is usually known as the *transformed system*.

The force term used depends on the state x of the canonical system and the target y_{ref}, and takes the form

$$f(x) = \alpha_w \frac{\sum_{n=1}^{N} \Psi_n(x) w_n x}{\sum_{n=1}^{N} \Psi_n(x)},$$

where each Ψ_n is a basis function, the w_n are adjustable weights, and α_w is a scaling factor. The basis functions used throughout this work consist of Gaussian kernels

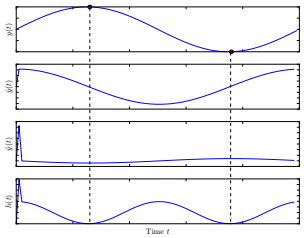
$$\Psi_n(x) = \exp\{-(t/T - c_n)/2\sigma_n^2\},$$

with c_n and σ_n^2 the center and width of the nth kernel.

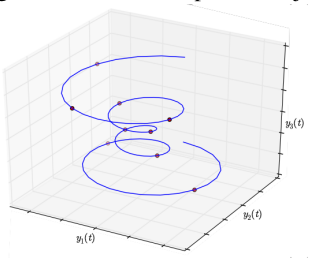
Finally, the transformed system can be augmented with an additional component used to accommodate (smooth) shifts in the target/reference position. Such component can be described as a first-order dynamical system

$$\dot{y}_{\text{ref}}(t) = \begin{cases} \frac{\Delta t}{T}(y_0 - y^*) & \text{if } t \leq T \\ 0 & \text{otherwise,} \end{cases} \qquad (2)$$

[2]The DMP formulation described and adopted throughout this work follows that of Kulvicius et al [12].



(a) Segmentation of a simple 1D trajectory.



(b) Segmentation of a 3D trajectory.

Fig. 3: Illustration of the segmentation module.

with y_0 and y^* representing the initial and final target positions.

In the continuation, we discuss how the different modules of our system are built on top of the DMP representation.

A. Segmentation

Given an initial demonstration $\{\mathbf{y}(t), t = 0, \ldots, T\}$ by a human user, we segment the trajectory into multiple segments, each of which is then turned into one DMP. To perform segmentation, we follow Meier et al [16] and identify segmentation points as minima in velocity and acceleration. In particular, letting

$$h(t) = \|\ddot{y}(t)\|^2 + \|\dot{y}(t)\|^2, \tag{3}$$

local minima of $h(t)$ are good candidates for segmentation points. Therefore, after computing $h(t)$ for all $t = 0, \ldots, T$, we compute local minima of $h(t)$ and mark those points as final points of a sub-trajectory and initial point of a new sub-trajectory. Fig. 3 illustrates the segmentation process with some simple trajectories.

After determining the set of sub-trajectories obtained from a demonstration, our system builds one DMP for each sub-trajectory. The learning of the DMPs consists of adjusting the weights of the force function so that the resulting trajectory matches the demonstration as closely as possible [27]. The different DMPs are then stored in a library for future use.

B. Optimization

Given the library of learned DMPs, whenever the user provides the robot with a new target, the robot must now use the learned DMPs to reach the new target while accommodating potential changes to the environment configuration (e.g., new obstacles, etc.). In order to do that, the robot identifies a sequence of DMPs that may meet the desired objectives (reaching the target, avoiding obstacles).

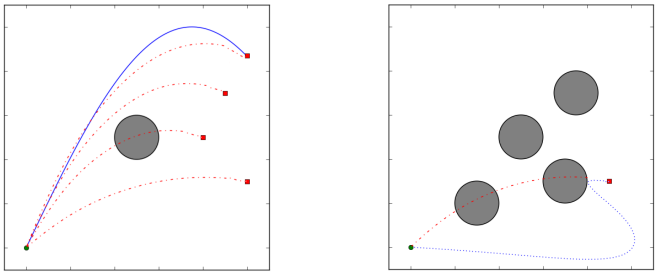
The identification of which DMPs can be used to reach a given target relies critically on the robot's ability to optimize a DMP to reach such target in light of environment constraints. We adopt the Covariance Matrix Adaptation Evolutionary Strategy—or CMA-ES—to that purpose. CMA-ES is a derivative-free optimizer that has been successfully used in robot learning and reinforcement learning [28], [25], [11].

Given a target position y^* and a set of M obstacles, $\{o_1, \ldots, o_M\}$, we define the objective function

$$J = \sum_{t=0}^{T} \Bigg[c_{\text{goal}} \|y(t) - y^*\|^2 + c_{\text{vel}} \|\dot{y}(t)\|^2 + c_{\text{obst}} \sum_{m=1}^{M} \mathbb{I}(y(t) \in o_m) \Bigg], \tag{4}$$

where c_{goal}, c_{vel} and c_{obst} weight the relative importance of reaching the goal, avoiding large velocities and avoid obstacles, and $\mathbb{I}(y(t) \in o_n)$ is a binary value that indicates whether, at time step t, the trajectory $y(t)$ hit obstacle o_n.

Given a learned DMP—encoded by an N-dimensional weight vector $\mathbf{w}$ that defines the force term in the DMP representation— CMA-ES proceeds at each iteration i by sampling a set of new candidate weight vectors, $\{\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(K)}\}$, from a multivariate Gaussian distribution centered around the current weight vector, $\mathbf{w}_i$, and with covariance $\mathbf{\Sigma}_i$. The trajectories resulting from each candidate weight vector $\mathbf{w}^{(k)}$ are evaluated according to the cost function J in (4), and a new weight vector $\mathbf{w}_{i+1}$ is computed from the weighted mean of the candidate vectors. The covariance matrix $\mathbf{\Sigma}_i$ is also adapted accordingly. Fig. 4a



(a) (b)

Fig. 4: Learning and optimization of simple DMPs from demonstration.The green circle and red square correspond to the start and end points, respectively.(a) Illustration of the result of learning a simple 2D DMP from demonstration. The solid line corresponds to the original demonstration, while the red-dashed lines correspond to the modulated DMP to different targets. (b) red-dashed line is a DMP trajectory modulated to the target. Since the DMP is unaware of the existence of obstacles, the modulated trajectory goes through the obstacle. The blue-pointed line corresponds to an optimized DMP modulated to a new target. CMA-ES is able to avoid the obstacle while reaching the target.

illustrates the result of the optimization process in a simple environment with an obstacle. The original DMP corresponds

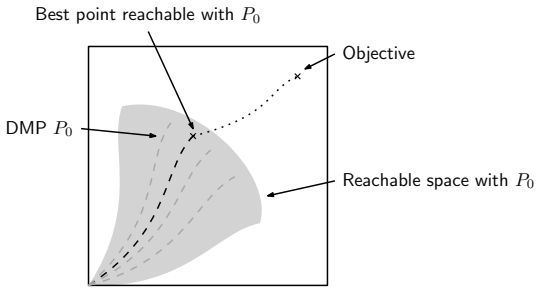


Fig. 5: Greedy approach to determining which DMPs to combine in order to achieve a new target while accommodating a new environment configuration. See text for details.

to a solid line, while the red-dashed lines correspond to successive trajectories obtained for different targets using the same DMP without optimizing. While in Fig. 4b the blue-pointed line success to avoid the obstacles and reach the target using optimizing by CMA-ES.

C. Combination

We now discuss the combination module in our framework. Such module can be broken down in two components:

- A first component that selects from the library the DMPs to be combined;
- A second component that adjusts the selected DMPs to obtain a single, smooth trajectory

We now describe each of these two components in detail.

1) Selecting the DMPs to combine: Our approach greedily selects a sequence of DMPs that drives the robot from an initial state y_0 to a target state y^* while respecting the constraints imposed by the environment as Fig. 5 illustrates. The selection process relies critically on the optimization approach described in Section III-B and works as follows.

Let y_0 denote the origin of the desired motion, and y^* the desired target. Using CMA-ES, we optimize each DMP in the library to move from y_0 to y^* while respecting the constraints imposed by the environment. The best DMP—in terms of the associated cost J—is then selected. If the target is reached the process concludes. Otherwise, the ending point of the selected DMP is set as the new origin and the process repeats until the target is reached.

2) Combining multiple DMPs into one smooth trajectory: Once a sequence of DMPs is identified that yields a combined trajectory from y_0 to the target y^*, it is necessary to combine these DMPs together to yield a novel trajectory, as smooth as possible. We adopt the approach of Kulvicius et al [12].

Given the sequence $\{m_1, \ldots, m_D\}$ of DMPs, selected as described in Section III-C.1, the combined trajectory will run for an extended time-frame

$$T' = \sum_{d=1}^{D} T_d,$$

where T_d is the length of the dth selected DMP. The forcing term for the combined trajectory is now composed from $N \times D$ basis functions, where basis function $\Psi_{n,d}$ is now a Gaussian kernel centered around

$$c_{n,d} = \begin{cases} \frac{T_1}{T'} \cdot \frac{n-1}{N-1} & \text{if } d = 1 \\ \frac{T_d}{T'} \cdot \frac{n-1}{N-1} + \frac{1}{T'} \sum_{\ell=1}^{d-1} T_\ell & \text{otherwise.} \end{cases}$$

with a width

$$\sigma_{n,d} = \sigma_n \frac{T_d}{T'},$$

where σ_n is the width of the original basis function Ψ_n. The extended set of basis functions now spans the whole time frame T' and the corresponding widths ensure some level of overlap which, in turn, leads to a smoother resulting trajectory. The weight associated with each basis function $\Psi_{n,d}$ is the nth weight of the dth DMP.

Finally, we also modify (2) to reflect the dynamics of the successive (intermediate) targets as

$$\dot{y}_{\text{ref}}(t) = \begin{cases} \frac{\Delta t}{T_d}(y_{0,d} - y_d^*) & \text{if } \sum_{\ell=1}^{d-1} T_\ell \leq t \leq \sum_{\ell=1}^{d} T_d, \\ 0 & \text{otherwise,} \end{cases}$$

where $y_{0,d}$ and y_d^* are the origin and target of the dth DMP.

The resulting combined motion is a novel (complex) DMP which can also, in turn, be optimized using the approach described in Section III-B.

D. Execution-time avoidance of moving obstacles

Finally, during trajectory execution, it is possible to adapt runtime obstacle-avoidance approaches [29]. Specifically, the dynamics of the system can be modified by a *dynamic modulation matrix* $\mathbf{M}$. Such matrix is computed taking into account observable properties of the obstacles and the environment, such as desired safety margin, tail-effect, and reactivity. Writing DMPs as single order dynamical systems, we get the following equivalent formulation

$$\begin{aligned} \dot{z}_0(t) &= -\frac{\alpha_x \exp\{\alpha_x(T-t)/\Delta t\}}{1 + \exp\{(\alpha_x(T-t)/\Delta t)^2\}}, \\ \dot{z}_1(t) &= z_2(t), \\ \dot{z}_2(t) &= \alpha_y(\beta_y(z_3(t) - z_1(t)) - z_2(t)) + f(z_0(t)), \\ \dot{z}_3(t) &= \begin{cases} \frac{\Delta t}{T_d}(y_{0,d} - y_d^*) & \text{if } \sum_{\ell=1}^{d-1} T_\ell \leq t \leq \sum_{\ell=1}^{d} T_d, \\ 0 & \text{otherwise,} \end{cases} \end{aligned} \tag{5}$$

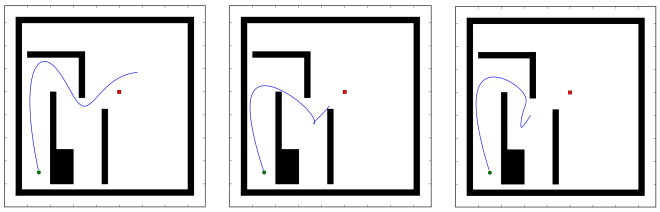
where $z_0 = x$, $z_1 = y$, $z_2 = \dot{y}$ and $z_3 = y_{\text{ref}}$. The dynamic modulation matrix can be computed at runtime to warp the trajectories away from the moving obstacles as

$$\dot{\mathbf{z}}(t) = \mathbf{M}(\mathbf{z})\mathbf{f}(\mathbf{z}(t), t),$$

where we wrote $\dot{\mathbf{z}}(t) = \mathbf{f}(\mathbf{z}(t), t)$ to compactly denote the system (5).

IV. Experiments

We now illustrate our framework both in simulated and real trajectory generation. Our results showcase the ability of our framework to generate complex motions from learned DMPs. We illustrate both the performance of the individual modules and of the system as a whole.



(a) (b) (c)

Fig. 6: Several runs of CMA-ES in a cluttered environment. The trajectories are generally unable to reach the target (the red square).

A. Learning simple DMPs from demonstration

We start by showcasing the process of learning a simple DMP from demonstration.

Fig. 4a illustrates the DMP resulting from a simple 2D demonstration. In the example, the dashed line corresponds to the DMP learned from the solid line. If we change the target of the DMP, the resulting trajectory will maintain the same overall shape, while ending in the desired target, but it will unable to avoid the obstacle.

We note, however, that although the DMP illustrated in Fig. 4a was learned in the presence of an obstacle (corresponding to the gray circle), it contains no information regarding such obstacle; therefore, if modulated to a target across the obstacle, the resulting trajectory will not avoid the obstacle, as illustrated in Fig. 4b.

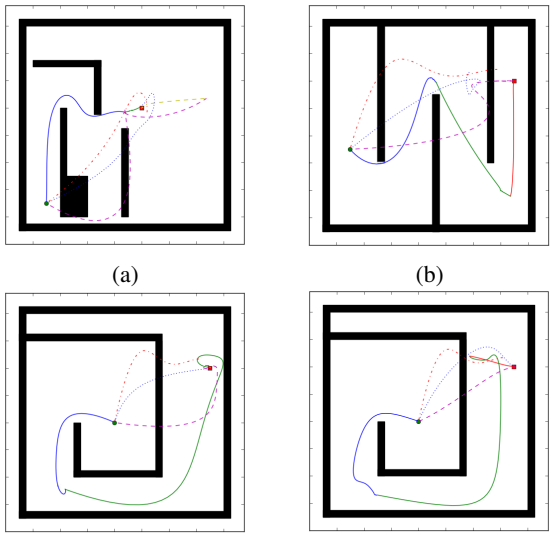
The use CMA-ES readily handles such situation. By including obstacle information in the cost function J, CMA-ES departs from the learned DMP and construct a new DMP trajectory that is able to successfully reach the target and avoid the obstacles, as illustrated in Fig. 4b.

B. Composition of DMPs for cluttered environments

Although CMA-ES is able, to some extent, to address the constraints imposed by the environment to the DMP trajectories, in very cluttered environments it may not be enough. Fig. 6 illustrates several cases in which CMA-ES was unable to attain the target position due to the constraints imposed by the environment. In contrast, our approach is able to successfully generate trajectories that reach the target while avoiding the obstacles, as illustrated in Fig. 7 and 10a. Fig. 7 illustrates the performance of each individual component effect on the framework as a whole, we can see how our framework has succeeded in all the difficult environments and different conditions, but the other ways couldn't reach the target each time, and they couldn't avoid the obstacles and get a safe path to the target.

We note that the combined trajectory can, in turn, be optimized (again with CMA-ES) to obtain a yet smoother trajectory. Fig. 9 showcases the trajectory resulting from applying CMA-ES to the combined trajectory .

Finally, we illustrate in Fig. 8 the dependence of our combination approach on the optimization module. In particular, we depict the resulting combinations when, during the selection stage, we run CMA-ES for different numbers of iterations and step-size. As evidenced in the plots, running CMA-ES for a larger number of generations allows for "straighter" trajectories. and by using a fit step-size provide good exploration for the environment..



(a) (b) (c) (d)

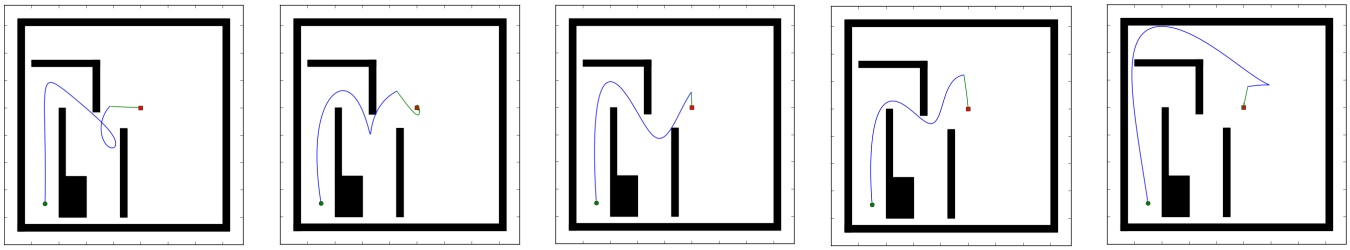
Fig. 7: Examples where our approach could successfully combine multiple DMPs to reach the target position in different environments and different conditions. The coloured line is execution by our framework using planning, segmentation and optimization, it depicts the different DMPs with different colours. The coloured-dashed line is a segmentation without optimization, sometimes we have over one sub-DMP. The red-dashed-pointed line is the normal DMP without optimization. The blue-pointed line is the DMP with optimization. The green circle and red square correspond to the start and end points, respectively. (d) The framework did not have enough iterations, and it needs three sub-DMPs.

C. Execution-time obstacle avoidance

During the execution of a trajectory, it may happen that the robot needs to accommodate a moving obstacle. For example, during the interacting with a human user, the user may move and the robot should be careful to avoid hitting the user. In other words, the robot should treat the user as a "moving obstacle" and adjust its motion to the motion of the human.

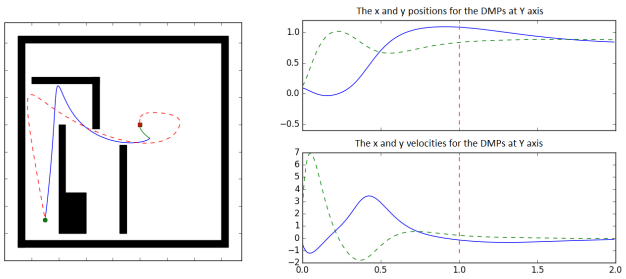
As discussed in Section III, our framework accommodates one such module, that addresses precisely the avoidance of obstacles at execution time. We illustrate in Fig. 11 a sequence of frames in which the trajectory executed by the robot is slowly adjusted to avoid the obstacle.

Finally, we also applied our framework in the Baxter robot, as illustrated in Fig. 10b. In the experiment, the robot must transfer the green object to the right side of the obstacle, having only been taught how to move its arm in the absence of obstacles. Once again, Baxter is able to successfully move the blue object without knocking down the obstacles.



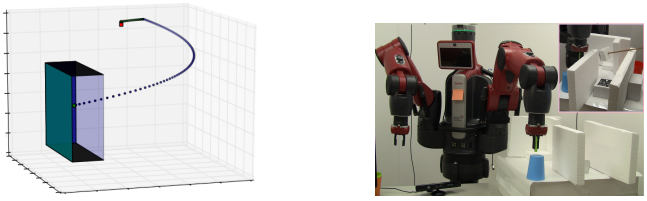
(a) 25 CMA-ES generations. (b) 50 CMA-ES generations. (c) 250 CMA-ES generations. (d) 300 CMA-ES generations. (e) CMA-ES has a big step-size.

Fig. 8: Dependence of the combined trajectory on the underlying optimization process.The green circle and red square correspond to the start and end points, respectively.



(a) (b)

Fig. 9: Application of CMA-ES to the combined trajectory.(a) Upon applying CMA-ES to the combined trajectory (depicted with a solid line), we obtain the significantly smoother dashed trajectory.The green circle and red square correspond to the start and end points, respectively. (b) Position and velocity profiles for the optimized combined trajectory. The dashed red line indicates the point at which the two original DMPs were put together.



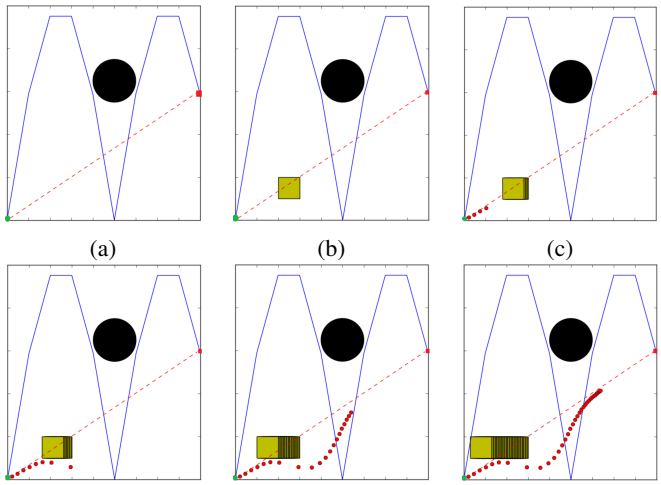
(a) (b)

Fig. 10: (a) Example of the application of our approach in a 3D environment with obstacles.The green circle and red square correspond to the start and end points, respectively. (b) Setup for the real-world experiment using the Baxter robot.

V. CONCLUSIONS

In this paper, we contributed an end-to-end framework to learning and generating complex motions from a library of "atomic" actions (DMPs) learned by decomposing a demonstration from a human expert. Our approach relies critically on the robot's ability to optimize a DMP's trajectory to reach a target while complying with the environment constraints. Our approach is also able to accommodate execution-time obstacle avoidance. Although our results involve relatively simple domains, they are, nevertheless, illustrative of the the potential of our proposed approach.

Our research also opens interesting avenues for future work. For example, it would be interesting to investigate the use our approach in tasks involving *prediction* and *recognition*, aimed at human-robot interaction tasks.



(a) (b) (c) (d) (e) (f)

Fig. 11: Sequence of frames showing execution-time obstacle avoidance. (a) The blue solid line corresponds to the demonstrated trajectory, from which several simple DMPs are learned to a library. The dashed line corresponds to the optimized trajectory from the bottom left to the target (on the right). (b) The square represents an obstacle that is moving leftwards. (c) and (d) The sequence of red dots corresponds to the actual trajectory executed by the robot. Note that, as soon as the obstacle is out of the way, the robot resumes its initially planned trajectory.The green circle and red square correspond to the start and end points, respectively.

ACKNOWLEDGMENT

This paper is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-19-1-0020 and by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UIDB/50021/2020. The first author acknowledges the Ph.D. grant from the Global Platform for Syrian Students.

REFERENCES

[1] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in Cognitive Sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[2] B. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[3] M. Lopes, F. S. Melo, L. Montesano, and J. Santos-Victor, "Abstraction levels for robotic imitation: Overview and computational approaches," in *From Motor Learning to Interaction Learning in Robots*, O. Sigaud and J. Peters, Eds. Springer, 2010, pp. 313–355.

[4] A. Billard, Y. Epars, S. Calinon, G. Cheng, and S. Schaal, "Discovering optimal imitation strategies," *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 69–77, 2004.

[5] R. Rao, A. Shon, and A. Meltzoff, "A Bayesian model of imitation in infants and robots," in *Imitation and social learning in robots, humans, and animals*. Cambridge University Press, 2007.

[6] G. Hayes and J. Demiris, "A robot controller using learning by imitation," in *Proc. 2nd Int. Symp. Intelligent Robotic Systems*, 1994, pp. 198–204.

[7] M. Asada, M. Ogino, S. Matsuyama, and J. Ooga, "Imitation learning based on visuo-somatic mapping," in *Proc. 9th Int. Symp. Experimental Robotics*, 2006, pp. 269–278.

[8] G. Maistros, Y. Marom, and G. Hayes, "Perception-action coupling via imitation and attention," in *Proc. AAAI Fall Symp. Anchoring Symbols to Sensor Data in Single and Multiple Robot Systems*, 2001.

[9] M. Matarić, "Sensory-motor primitives as a basis for learning by imitation: Linking perception to action and biology to robotics," in *Imitation in Animals and Artifacts*, 2002.

[10] A. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013.

[11] F. Stulp and O. Sigaud, "Path integral policy improvement with covariance matrix adaptation," in *Proc. 29th Int. Conf. Machine Learning*, 2012, pp. 1547–1554.

[12] T. Kulvicius, K. Ning, M. Tamosiunaite, and F. Worgötter, "Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting," *IEEE Trans. Robotics*, vol. 28, no. 1, pp. 145–157, 2012.

[13] S. Khansari-Zadeh and A. Billard, "A dynamical system approach to real-time obstacle avoidance," *Autonomous Robots*, vol. 32, no. 4, pp. 433–454, 2012.

[14] K. Zadeh, S. Mohammad, and A. Billard, "Realtime avoidance of fast moving objects: A dynamical system-based approach," in *Electronic proc. of the Workshop on Robot Motion Planning: Online, Reactive, and in Real-Time [IROS" 2012]*, no. EPFL-POSTER-181892, 2012.

[15] F. Meier, E. Theodorou, F. Stulp, and S. Schaal, "Movement segmentation using a primitive library," in *Proc. 2011 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2011, pp. 3407–3412.

[16] F. Meier, E. Theodorou, and S. Schaal, "Movement segmentation and recognition for imitation learning," in *Artificial Intelligence and Statistics*, 2012, pp. 761–769.

[17] S. Niekum, S. Osentoski, G. Konidaris, and A. Barto, "Learning and generalization of complex tasks from unstructured demonstrations," in *Proc. 2012 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2012, pp. 5239–5246.

[18] K. Mulling, J. Kober, and J. Peters, "Learning table tennis with a mixture of motor primitives," in *Proc. 2010 IEEE-RAS Int. Conf. Humanoid Robots*, 2010, pp. 411–416.

[19] B. Nemec, M. Tamosiunaite, F. Woergoetter, and A. Ude, "Task adaptation through exploration and action sequencing," in *Proc. 9th IEEE-RAS Int. Conf. Humanoid Robots*, 2009, pp. 610–616.

[20] J. Kober and J. Peters, "Policy search for motor primitives in robotics," *Machine Learning*, vol. 84, pp. 171–203, 2011.

[21] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *J. Machine Learning Research*, vol. 11, pp. 3137–3181, 2010.

[22] F. Sehnke, C. Osendorfer, T. Rückstiess, A. Graves, J. Peters, and J. Schmidhuber, "Parameter-exploring policy gradients," *Neural Networks*, vol. 23, no. 4, pp. 551–559, 2010.

[23] V. Heidrich-Meisner and C. Igel, "Evolution strategies for direct policy search," in *Proc. 10th Int. Conf. Parallel Problem Solving from Nature*, 2008, pp. 428–437.

[24] T. Rückstiess, F. Sehnke, T. Schaul, D. Wierstra, Y. Sun, and J. Schmidhuber, "Exploring parameter space in reinforcement learning," *Paladyn J. Behavioral Robotics*, vol. 1, pp. 14–24, 2010.

[25] F. Stulp and O. Sigaud, "Robot skill learning: From reinforcement learning to evolution strategies," *PALADYN Journal of Behavioral Robotics*, vol. 4, no. 1, pp. 49–61, 2013.

[26] A. Abdolmaleki, "Information theoretic stochastic search," Ph.D. dissertation, Universities of Minho, Aveiro and Porto, December 2018.

[27] S. Manschitz, J. Kober, M. Gienger, and J. Peters, "Learning movement primitive attractor goals and sequential skills from kinesthetic demonstrations," *Robotics and Autonomous Systems*, vol. 74, pp. 97–107, 2015.

[28] I. Loshchilov, "A computationally efficient limited memory cma-es for large scale optimization," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2014, pp. 397–404.

[29] K. Zadeh, S. Mohammad, and A. Billard, "A dynamical system approach to realtime obstacle avoidance," *Autonomous Robots*, vol. 32, no. 4, pp. 433–454, 2012.